Dokumentáció

**Specifikáció:**

Amikor a játékos elindítja a programot, be kell írnia a nevét és utána a főmenüvel találja szemben magát, ahol menüpontok közül választhat:

1. Játék indítása: elindul a játék az adott beállításokkal
2. Pályaméret: a játékos beállíthatja, hogy hányszor hányas pályán szeretne játszani (max:40x40, min:20x20)
3. Nehézség: ebben a menüpontban kiválasztja a játékos, hogy milyen gyorsan menjen a kígyó, és hány plusz fal legyen

Ezek alapján lesz eldöntve a pontok felszorzása (egy pont, ha a kígyó megesz egy almát és ennek vannak szorzói a nehézségtől függően)

1. Bezárás: a program megáll

A játék:

A kígyót a WASD-vel irányítja a játékos. A falakat #-ek fogják szimbolizálni, a járható pályát pontok és a kígyót meg egy X fej és O test. Az almát egy A betű. Az akadályok a következők lesznek:

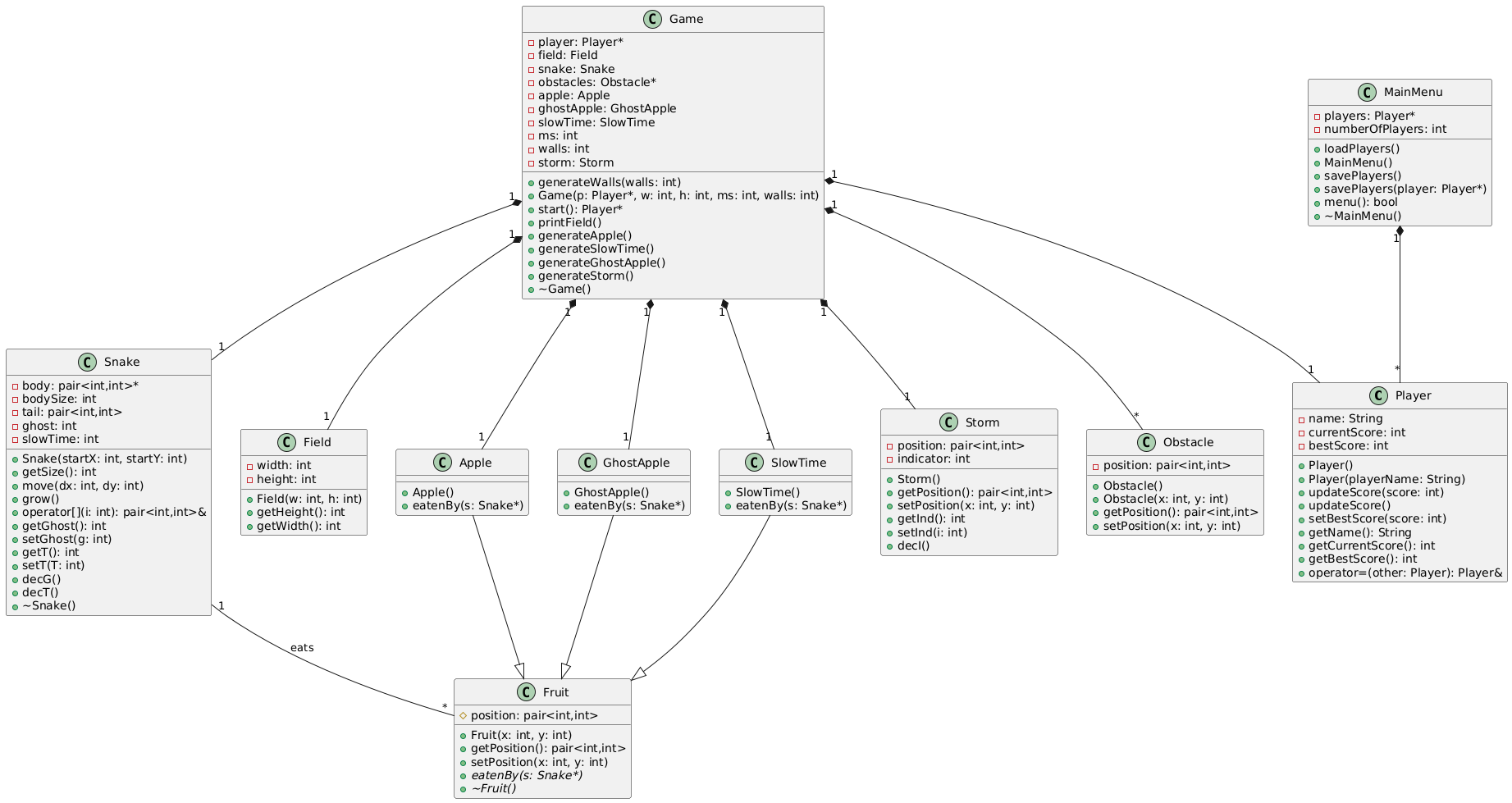
1. Vihar: a pályán random elkezd felerősödni egy vihar, ami előre látszik és nem veszélyes, de egy kis idő múlva az lesz. Ez a vihar homok színű és a pályán egy 3x3-as sávot fed le.
2. Több fal: a pálya közepén random falak fognak megjelenni, amennyit beállít a játékos (minél többet állít be, annál nagyobb a szorzó)

A power-uppok a következők:

1. Idő lassítás: Egy T betű, amire, ha rámegy a kígyónk, belassul az idő (lassabb játékoknál lehet, hogy idegesítő, de gyorsabbaknál nagyon hasznos tud lenni)
2. Átlátszóság alma: ha ezt a G betűt megeszi a játékos, a kígyója áttűnőbb lesz egy kis ideig, és ekkor át tud menni a falakon, ráadásul saját magán is át tud kúszni

A játékban bal alsó sarkon lesz kiírva a mostani pontszám, és a jobb alsó sarokban, pedig az eddigi legjobb eredményed. A játékos bármikor bezárhatja a játékot, és visszaléphet a főmenübe az ESC-el, és automatikusan elmenti a játékos nevét és eredményét egy külső fájlba.

**Terv:**



a végső tervben a Game class fogja irányítani az egész játékot, amit a MainMenu példányosít

**Függvények:**

MainMenu:

1. loadPlayers: beolvassa a txt fájlból a neveket és a legjobb eredményeket, és dinamikusan tárolja
2. MainMenu: beállítja az alapértékeket, és köztük beolvastatja a fájlt
3. savePlayers: kiírja fájlba a játékosokat, ez kap egy player pointer,t hogy el tudja menteni az updatelt score-ját
4. menu: ez maga a menü, ami kezeli az opciók kiírását, és a beállításokat, valamint ennek van egy bool visszatérési érteke, és ez attól függ, hogy a játékos kilépett e vagy nem, egyben ez idítja el a játékot is
5. a destruktora pedig a dinamikusan foglalt helyet kitörli

Game

1. a getLatestKey és a resetCursorPosition csak a kiírásokért, és az optimális wasd-ért felelősek, úgyanígy, a random is magától érthetődő
2. start: ennek van egy Player pointer visszatérési értéke, hogy ha megdöntötte a rekordját, akkor el tudjuk menteni, a while ciklusában először is ellenőrzi, hogy meghalt-e a kígyónk, utána, hogy milyen almákat ehettünk meg, majd a mozgás inputot ellenőrzi
3. printField: kiírja a pályát
4. Generate függvények: random legenerálják a pályára a dolgokat, figyelve a kígyóra, hogy ne ütközzenek vele

Player

Ez az osztály egy nevet, egy mostani scoret és egy best scoret tárol

1. updateScore: attól függően, hogy van-e paramétere, vagy nincs tud resetelni és tud növelni a current score-on
2. = operátor

Snake

1. resize: csak a grow függvény használja, ez dinamikusan hozzáad egy elemet a tömbhöz
2. Snake: alapra állítja a snake-et
3. [] operátor: a body-nak az adott elemét adja vissza
4. move: a tail változó arra kell, hogy ha megeszik egy almát, annak a helyére spawnoljon az új testrész, ezen kívül, a body pozíciókat az előtte lévő bodyra cseréli le, így mozog a kígyó
5. grow: megnő a kígyó
6. destruktor: a dinamikusan foglalt body-t kitörli

Targy:

1. Ezeket a Game kezeli, az eatenBy függvény csak meghívja a snake-nek vagy a grow()-ját vagy beállítja a tulajdonságait